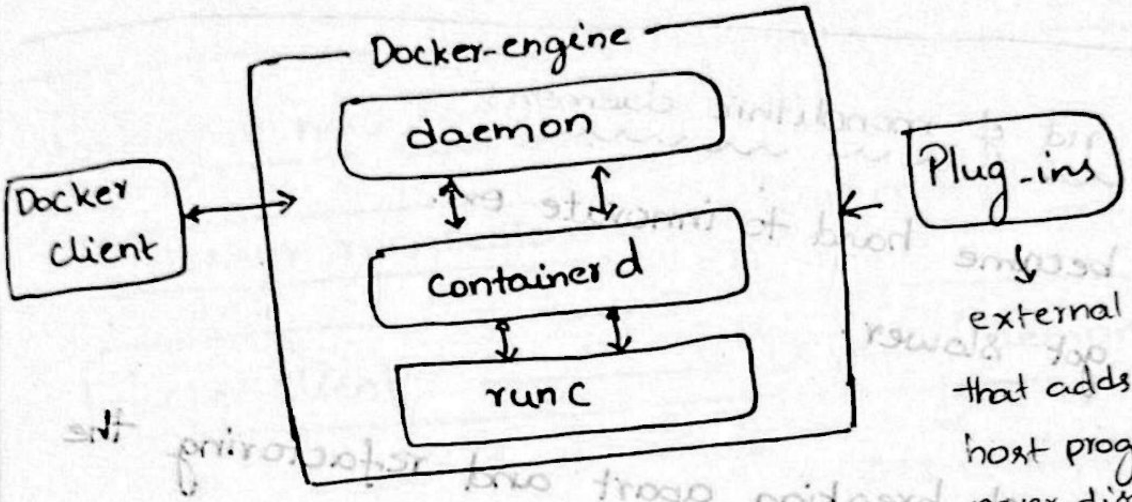
Docker Engine:-

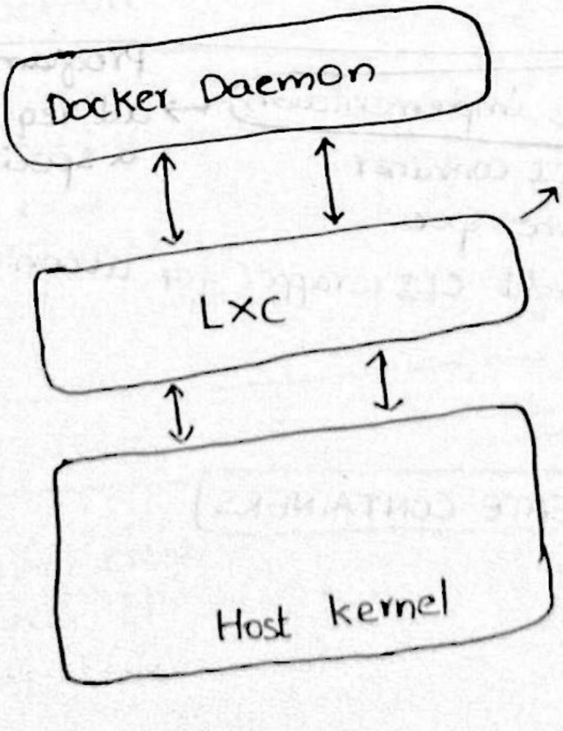
↳ Core Software that runs & manages containers.

↳ modular in design & built from many small Specialized tools.



↳ external software that adds fns to host program but never disturbs the host program.

Initial docker architecture:- ↗ monolithic binary does all the activities todays docker-engine does.



↗ Provided access to fundamental building parts of container in the OS kernel

Getting Rid of LXC :-

- LXC is Linux-specific ∴ Problem for docker to be multiplatform.
- So, Docker Inc developed "libcontainer" and it is the default execution driver since "0.9" releas

Getting rid of monolithic daemon :-

- It became hard to innovate on.
- It got slower.

The work of breaking apart and refactoring the Docker engine has seen all of the Container execution & runtime code entirely removed from daemon.
&
refactored into Small, specialized tools.

runc → reference implementaion of the OCI container run-time spec → Program that implem all requirements of a specification.

→ Small, lightweight CLI wrapper for libcontainer

→ Sole purpose → CREATE CONTAINERS

Container execution logic

container d → ⇢ Sole Purpose → manage container lifecycle operations ~~run~~, start, stop. pause, rm

---

Starting a new Container :-

docker run container on docker client

docker Client → Docker Daemon

↓ Command is converted into appropriate API payload and POSTed to API endpoint exposed by "daemon"

↓ API is implemented in daemon

Docker Daemon → containerd

↓ If the command is to create a new container, it makes a call to containerd

↓ Communication VIA CRUDE style API over gPRC.

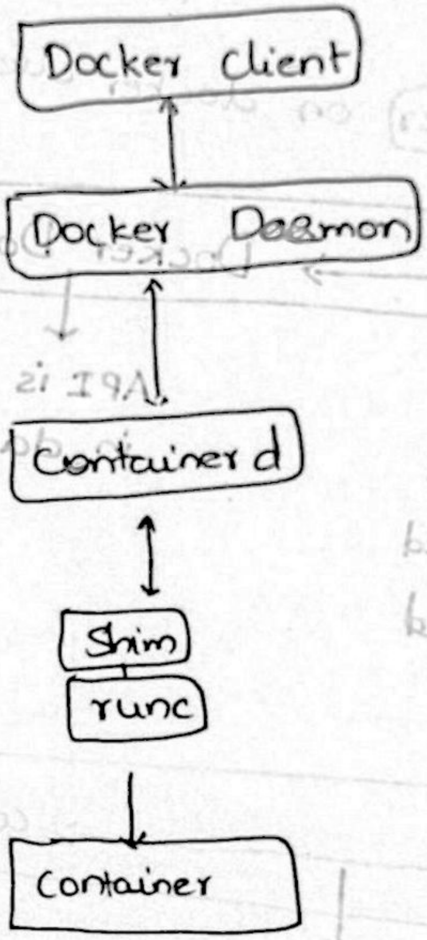
Containerd → run c

↓ containerd cannot create new containers. It converts image to OCI bundle and forwards to runc

runc

* Interfaces with OS kernel to pull together all resources required to create a container.

* Container process started as a child process of runc, as soon as it started runc will exit.



what is shim?

Once a container's parent runc process exits, associated containerd-shim process becomes the parent.

Having all logic to start & run containers removed from the daemon means

Entire Container runtime decoupled from Daemon.

↕

Daemonless containers

↓

So, could start & stop daemon at will without disturbing Container runtime.

Client & Daemon Connection:-

Docker implements client & server model

client ↓ Docker client

server ↓ Daemon.

Default installation puts them on the same host & configures them to communicate over a local IPC socket.

* possible to configure them to communicate over a network.

Communication over an unsecured HTTP Socket on port 2375/tcp